Voorlopig

THE-RC 41523

Technische Hogeschool Eindhoven
Rekencentrum

RC-Informatie AG-25
40 bladzijden
datum: augustus 1980

Handleiding CPU-4
Single Board Microcomputer

Inhoudsopgave:

	bladzijde
1. Inleiding CPU-4	3
2. TIM, Teletype Interface Monitor	4
2.1. TIM-commando's	4
2.2. Opstarten TIM	5
2.3. Interrupt/reset	6
2.4. TIM I/O-subroutines voor TTY	6
2.5. TIM geheugengebruik	7
3. Resident assembler	8
3.1. Input line format	8
3.2. Assembler karakterset	8
3.3. Operands	8
3.4. Assembler commando's	9
3.5. Labeldefinitie	10
3.6. Foutmeldingen	10
3.7. Voorbeelden	11

4. TINY BASIC 13
4.1. Format 13
4.2. Operators 13
4.3. Expressions 14
4.4. Intrinsieke functies 15
4.5. Statements 16
4.6. Fouten 19
4.7. Definities 21
4.8. Voorbeelden 22
5. Geheugengebruik 27
6. CPU 6502-4 hardware 29

Auteurs: T. van der Meer
W. Vierbergen
H. Wijtvliet
(groep CS)

1. Inleiding CPU-4.

CPU-4 is een single board computer die ontworpen is voor applicaties die gemaakt zijn op het PUZZLE-ontwikkelingssysteem. Beiden hebben dezelfde busstructuur en kunnen daarom dezelfe I/O- of geheugenkaarten gebruiken als expansiemogelijkheid.

Eigenschappen:

Processor	6502
EPROM	4K
RAM	2K
I/O	10 TTL bidirectioneel
TTY	V24 (tot 600 baud)
Power supply	5 Volt
Data/adreslijnen	gebufferd
Klok	1 MHz
Monitorprogramma (TIM)	1K ROM + timmer

Als applicatie wordt hierna de beschrijving gegeven van integer TINY BASIC (startadres $C000) en een one pass assembler (startadres $C900). Bovendien wordt een beknopt overzicht gegeven van het monitorprogramma. De 10 beschikbare I/O-lijnen kunnen gebruikt worden om een high speed tape reader te sturen; het monitorprogramma biedt hiervoor ook de faciliteiten.

2. TIM, Teletype Interface Monitorprogramma.

2.1. TIM-commando's.

(Onderstreepte karakters worden door de gebruiker ingetypt.)

CR	Carriage return na een reset bepaalt de teletype-snelheid.
.R	Laat de inhoud van de registers zien: Format: PC-P-A-X-Y-S PC Program counter P Process status A Accumulator X, Y Registers S Stack pointer.
.G	Go, start een programma dat begint op lokatie PC dat ingevuld is door een R-commando.
.M addr	Laat de inhoud zien van 8 geheugenplaatsen, beginnend bij adres addr.
.: ADDR data	Geheugen of registers veranderen, afhankelijk of de voorgaande instructie een R- of M-commando was. Velden die niet veranderd hoeven te worden, kunnen geskipt worden door een spatie in plaats van data op te geven of ongewijzigd blijven door CR in te typen. Het geheugen wordt dan automatisch verhoogd. TIM geeft tussenliggende spaties.
.WH addl addh CR	Papertape dump van het geheugen. Vanaf addl tot addh volgt een hexadecimaal format: ;ccaaaaddd...ddsss cc Aantal bytes aaaa Adres waar de volgende bytes vandaan komen ddd... Data, hexadecimaal (2 digits/byte) ssss Checksom van de hele regel. Eventueel afsluiten met ;00 CR.

.LH	Laadt papertape met hexadecimaal format dat afgesloten is met ;00 CR. Bij foutmelding: "?".
.WB	Papertape dump volgens BNPF-format. aaaa BdddddddF Bd..F Bd..F Bd..F aaaa Adres van de eerste 4 bytes B "B" begin byte F "F" einde byte ddd "P" true "N" false.
.LB	Papertape met BNPF laden.

2.2. Opstarten TIM.

Onderstreepte waarden intikken, de rest antwoordt TIM.
CR = Carriage return

1) Reset knop indrukken
2) CR
3) * 7052 XX XX XX XX
4) . M00EA XX XX ...
5) . : 00EA 11 11 CR
6) . R 7052 XX .....
7) . : C000 CR Startadres TINY BASIC
8) of : C003 CR Restartadres TINY BASIC
9) of : C900 CR Start assembler
10) . G

1) en 2) Initialiseert het systeem en alle interruptvectoren. Tevens wordt automatisch de baudrate ingesteld van de teletype.

4), 5) en 6) Alléén gebruiken bij papertape dump of read. Hier worden de bit delay registers $EA en $EB gevuld met respectievelijk $11 en $11 voor nauwkeurige instelling van de baudrate van een mechanische ASR-390 teletype op 110 baud. Deze werkt op 2 stopbits en TIM op 1 stopbit.

2.3. Interrupt/reset.

Interrupt.

Alle interruptvectoren worden na een reset geïnitialiseerd en wijzen dan naar het monitorprogramma.

- Software break (BRK = $00).
  De vector hiervoor staat in lokatie $FFFE en FFFF. Als deze vector niet door de gebruikers veranderd wordt, reageert TIM met een asteriks < * > en de inhoud van de registers (zie R-commando).
  (N.B. Break is een 2-byte instructie.)
- Hardware break (IRQ-pin).
  De vector-stand is lokatie $FFFE en FFFF. Als uit het statusregister blijkt dat deze interrupt géén BRK was, kan er gesprongen worden naar een interrupt service routine waarvan de vector staat in lokatie $FFF8 en $FFF9. Door TIM geïnitaliseerd wordt er geantwoord met < # * registerinhoud >.
- NMI.
  Vectoren in lokatie $FFFA en FFFB.
  TIM antwoordt met < # * registerinhoud >.

Reset.

Na een reset initialiseert TIM naar zichzelf, zet alle defaultvectoren en wacht op een carriage return van de TTY om zijn baudrate te bepalen. De resetvector is niet door de gebruiker te beïnvloeden.

2.4. TIM I/O-subroutines voor TTY.

Subroutine naam	Adres $	Handeling	Arg.	Result	Veranderde registers
WRT	72C6	karakter output	Acc	--	Acc, X
RDT	72E9	karakter input	--	Acc	X, Y
CRLF	728A	CR-LF en delay	--	--	Acc, X
SPACE	7377	spatie-output	--	--	--
WROB	72B1	print een byte hexadecimaal	Acc	--	Acc, X
RDHSR	733D	niet van toepassing	--	?	--

2.5. TIM geheugengebruik.

$7000 -	$7400	TIM ROM 1 Kbytes
$FF00 -	$FFFF	TIM RAM 64 bytes
$00E3 -	$00FF	Page zero gebruik
	$E3	CR-LF delay in bit-times
$EA -	$EB	Bit-time (baudrate)
UINT	$FFF8/9	User IRQ-vector
NMI	$FFFA/B	Hardware NMI-vector
RESET	$FFFC/D	Hardware reset-vector
IRQ	$FFFE/F	Hardware IRQ/BRK-vector

3. Resident assembler.

De assembler is ontwikkeld voor het vertalen van assemblertaal in object code, het toekennen van adressen aan labels en het bijhouden van een symbol table. Aangezien niet wordt uitgegaan van een editor source text, wordt iedere regel meteen vertaald in object code waarbij de symbol table zonodig updated wordt. Na een < .END >-operand wordt de symbol table in het object code-geheugen ingevuld en voorzien van een eindlisting. Assembleren kan dus maar éénmalig en alleen vanaf het keyboard.

3.1. Input line format.

Labels	Beginnen in kolom 1 en worden met een spatie afgesloten.
Instructie of pseudo instructie	Eerst een spatie, dan één van de 650X-assembler mnomonics of assembler commando's.
Operands	Eerst een spatie, dan één van de vormen beschreven bij operands.
Comments	Eerst een < ; > in kolom 1 of anders voorafgegaan door spatie.

Alle regels worden met een CR afgesloten.

3.2. Assembler karakterset.

Letters	A tot en met Z
Nummers	0 tot en met 9
Naamkarakter	. / : @ < = >
Andere karakters	spatie # $ ' ; ,

3.3. Operands.

- Namen.

  Een naam is ieder alfanumeriek symbool en wordt gebruikt als label of in operand fields. Het eerste karakter van iedere naam kan een letter of een naamkarakter zijn, de volgende mogen ook nummers zijn. Lengte is 'onbeperkt'.

  Bijvoorbeeld: DIT.IS.EEN.NAAM
  =NAME
  LABEL

- Nummers.

  Decimaal 0 tot en met 9

  Hexadecimaal 0 tot en met 9 en A tot en met F, voorafgegaan door < $ >.

  Bijvoorbeeld: $FFD
  275
  $40

  (Alleen de laatste 16 bits van een getal worden verwerkt.)

- Ascii-strings.

  In combinatie met immediate operands kunnen ascii-strings bestaan uit één ascii-karakter. Ze moeten dan door een < ' > worden voorafgegaan.

  Bijvoorbeeld: 'A
  '$
  'DIT IS EEN STRING'

3.4. Assembler commando's.

Deze geven geen directe object code maar worden gebruikt als hulp voor de assembler. Ze moeten door een spatie worden voorafgegaan.

.BYTE

Dit commando wordt gebruikt om het geheugen van te voren met vastgestelde bytes te vullen.

Bijvoorbeeld:

.BYTE $5F,37	Geeft $5F, $37
.BYTE 3,'ASC',$FC	Geeft $03, $41, $53, $41, $FC
.BYTE 'A,$AB	Is fout
.BYTE WRT	Geeft het lage order adres van de label WRT.

.WORD

Hiermee worden dubbele bytes (low, high) in het geheugen gevuld met 16-bits operands die van te voren zijn gedefinieerd.

Bijvoorbeeld als WRT = $72C6:

.WORD WRT	Geeft $C6, $72
.WORD $ADCD	Geeft $CD, $AD.

.END

Stopt het assembler-proces en geeft een listing van de symbol table.

.ORG

Geeft het startadres aan waar de object code geplaatst moet worden.

Bijvoorbeeld:

.ORG $0200	Programma begint vanaf lokatie $200
.ORG $START	START moet vooraf zijn gedefinieerd.

.RES

Dit commando reserveert een opgegeven aantal bytes in het geheugen, doet er verder niets mee.

Bijvoorbeeld:

.RES 15	Reserveert 15 (decimale) bytes.

3.5. Labeldefinitie.

label = operand

Bijvoorbeeld: FIVE = 5

3.6. Foutmeldingen.

R	Overflow branch adres
U	Verkeerde instructie-code
S	Operand code groter dan 8 bits
M	Hetzelfde label komt meerdere keren voor.

Verkeerd ingetikte regels kunnen hersteld worden door < \ > (shift L) te geven in plaats van carriage return.

3.7. Voorbeelden.

```
                ;CHECKOUT PROGRAM -- PRINT BINARY OF TYPED CHARACTER
                ;
                ;
0000                    *=0                     ;VARIABLE STORAGE IN PAGE ZERO
0000            BINARY  *=*+1                   ;STORAGE FOR CHAR DURING DISSE
0001            COUNT   *=*+1                   ;COUNT OF BITS REMAINING TO PR
                ;
0002                    *=$01C0                 ;PROGRAM BEGINS ON PAGE ONE
                ;
                CRLF    =$728A                   ;TIM  CRLF ROUTINE
                WRT     =$72C6                   ;TIM  WRITE ROUTINE
                RDT     =$72E9                   ;TIM  READ ROUTINE
                SPACE   =$7277                   ;TIM  SPACE ROUTINE
                ;
  C0  20 8A 72  PBIN    JSR CRLF                ;PRINT CARRIAGE RETURN & LINE F
01C3  20 E9 72          JSR RDT                 ;GET A CHARACTER
01C6  85 00             STA BINARY              ;SAVE FOR DISSECTION
01C8  20 77 72          JSR SPACE               ;PRINT A SPACE
                ;
01CB  A9 08             LDA #8                  ;INITIALIZE BIT COUNT
01CD  85 01             STA COUNT
                ;
01CF  A9 30     PBLOOP  LDA #'0                 ;ASSUME ZERO: LOAD ASCII "0"
0111  06 00             ASL BINARY              ;C=NEXT BIT
0113  90 02             BCC PRINT               ;PRINT ZERO
                ;
0115  A9 31             LDA #'1                 ;LOAD ASCII "1"
                ;
0117  20 C6 72  PRINT   JSR WRT                 ;PRINT BINARY DIGIT
011A  C6 01             DEC COUNT               ;COUNT BIT PRINTED
011C  D0 F1             BNE PBLOOP              ;DO NEXT BIT
                ;
011E  4C 00 01          JMP PBIN                ;DO IT ALL AGAIN
```

```
 *  7052 31 C0 FF 01 FF          Appendix B Revised  May 14, 1976
.I  E800
.G
                    ;       JOLT RESIDENT ASSEMBLER EXAMPLE
                    ;       ORG PSEUDO OPERATION
0000                        .ORG  $1F
001F                Z       .RES  0
001F                        .ORG  $1000
                    ;       IMMEDIATE OPERAND TYPES
1000  A9 2A         ABS     LDA  #42        ;DECIMAL
1002  69 2A                 ADC  #$2A       ; HEX
1004  69 2A                 ADC  #'*        ; ASCII
                    ;       ABSOLUTE OPERAND TYPES
1006  6D 0010               ADC  ABS        ; ABSOLUTE
1009  7D 0010               ADC  ABS,X      ; ABSOLUTE,X
100C  79 0010               ADC  ABS,Y      ; ABSOLUTE,Y
                    ;       INDIRECT OPERAND TYPES
100F  61 1F                 ADC  (Z,X)      ; INDIRECT,X
1011  71 1F                 ADC  (Z),Y      ; INDIRECT,Y
                    ;       ZERO PAGE OPERAND TYPES
1013  65 1F                 ADC  Z          ; ZERO PAGE
1015  75 1F                 ADC  Z,X        ; ZERO PAGE,X
                    ;       OTHER OPERAND TYPES
1017  0A                    ASL  A          ; ACCUMULATOR
1018  60            BACK    RTS             ; IMPLIED
                    ;       BRANCHES
1019  D0 FD                 BNE  BACK       ; BACKWARD
101B  D0 00                 BNE  FORWARD    ; FORWARD
                    ;       JUMPS
101D  CA            FORWARD DEX
101E  4C 1810               JMP  BACK       ; ABSOLUTE
1021  6C 0010               JMP  (ABS)      ;INDDREECT
                    ;       FORWARD REFERANCES
1024  0210                  .WORD FOR1
                    ;       UNDEFINED REFERANCE
1026  0210          FOR1    .WORD UNDEFINED
                    ;       PSEUDO OPERATIONS
1028                        .RES  4                     ; RESERVE STORAGE
102C  050F1F                .BYTE 5,$F,Z,'STRING'       ; DEFINE BYTES
1035                TEN     =     10                    ; EQUATE
1035  0A00CD                .WORD TEN,$ABCD,6500,256    ; DEFINE WORDS
                    ;       ERRORS
103D  C6 1F       M TEN     DEC  Z          ; MULTIPLY DEFINED SYMBOL
103F  71 00       S         ADC  (ABS),Y    ; OPERAND SIZE TOO LARGE
1041  DA 0010     O         DEC  ABS,Y      ; IMPROPER OPERAND FIELD
1044  00          U         DCC  Z          ; IMPROPER OPERATION CODE
1045  D0 D8       R         BNE  Z          ; BRANCH OUT OF RANGE
                    ;       THE SYMBOL TABLE WILL BE PRINTED AFTER
                    ;       THE END PSEUDO OPERATION.
                    ;       THE LOCATION OR VALUE OF THE SYMBOL IS
                    ;       PRINTED FOLLOWED BY THE SYMBOL NAME.
1047                        .END
  1000  ABS
  1018  BACK
  101D  FORWARD
  1026  FOR1
  000A  TEN
1026  UNDEFINED
  001F  Z

 *  EE13 32 00 00 00 F3
.WH 1000 1046
;181000A92A692A692A6D00107D00107900106 11F711F651F751F0A05E6
;18101860D0FDD000CA4C18106C00102610021010 54FE70050F1F530797
;171030545249 4E470A00CDAB64190001C61F7100DA001000D0D807C3
.
```

4. TINY BASIC.

In tegenstelling tot standaard extended BASIC kent TINY BASIC géén goniometrische en exponentiële functies, geen arrays en strings en geen floating point aritmetiek.

BASIC start op lokatie C000, de teletype input- en outputroutines liggen reeds vast. Indien het programma reeds aanwezig is (warme start), kan begonnen worden op lokatie C003. Hierbij moet het adresbereik van het programma (lokatie 20-23) vastliggen.
De backspace code (karakter delete) is SHIFT-O en de cancel code (line delete) is CONTROL-X.
Verdere bijzonderheden staan bij de statement-omschrijvingen.
Programma's in BASIC kunnen via de monitor papertape dump en load instructie worden bewaard. Opstarten gebeurt dan via lokatie $C003.
BASIC-programma's worden in het geheugen afgesloten met twee $00-bytes, het resterende geheugen wordt gebruikt voor het bewaren van return adressen in GOSUB-statements. Parameters en bewerkingen gebeuren in page zero.

4.1. Format.

Spaties in en meteen na een regelnummer worden weggegooid.
Regelnummering van 1 tot 32767.
Bijvoorbeeld:

```
123    PRINT "HELLO"
456    G O T O 1 2 3
7 8 9  PRINT "DIT IS LIJN =789"
```

Als een regel niet voorafgegaan wordt door een regelnummer, wordt de statement direct uitgevoerd. Op fouten wordt alleen getest op het moment dat de statements worden uitgevoerd.

4.2. Operators.

Op volgorde van afhandeling:

+	optellen
-	aftrekken
*	vermenigvuldigen
/	delen.

4.3. Expressions.

Alle gebruikte getallen worden herleid tot 15-bits binaire getallen met een tekenbit. Spaties hebben geen invloed en op overflow wordt niet getest. Komma's mogen niet voorkomen.

Bereik: -32768 tot 32767.

Bijvoorbeeld:

```
0
100
10 000          ; = 10000
1 2 3 4         ; = 1234
32767           ; = +32767
32769           ; = -32767
65536           ; = 0
65 636          ; = 100
```

Variabelen mogen alleen de hoofdletter A tot en met Z zijn.

Bijvoorbeeld:

```
A
123
1+2-3
B-14*C
(A+B)/(C+D)
-128/(-32768+(I*I))
(((((Q)))))
```

N.B. De volgende berekeningen geven allemaal dezelfde uitkomst:

Bijvoorbeeld:

```
-4096
15*4096
32768/8
30720+30720
```

4.4. Intrinsieke functies.

RND (bereik).

Deze functie geeft een positief random getal tussen nul en het opgegeven maximum bereik (bereik = 0 geeft een foutmelding).

USR (adres, val 1, val 2).

Het gebruik van machinetaal subroutines in BASIC waarbij:

adres = decimale adres van de subroutine (16 bits)

val 1 = 16-bits decimaal woord waarvan de linker 8 bits (most significant) het X-register worden

val 2 = 16-bits decimaal woord waarvan de rechter 8 bits (least significant) de accumulator worden.

val 1 en val 2 gelden voor zowel naar als van de subroutine. De subroutine moet afgesloten worden met een RTS-operand.

Bijvoorbeeld:

29382 = $72C6 (WRT), 66 = $42 (< B >)

A = USR(29382,0,66)

Er zal nu het karakter < B > via subroutine WRT op de terminal verschijnen.

Subroutines voor datatransport van en naar geheugenlokaties zijn reeds door TINY BASIC opgenomen:

Naar het geheugen:

adres = 37380 ($C018)

val 1 = geheugenadres, decimaal

val 2 = weg te schrijven waarde, decimaal.

Van het geheugen:

adres = 37376 ($C014)

val 1 = te lezen geheugenadres, decimaal

val 2 = krijgt de inhoud van het geheugenadres.

Terug naar monitor:

A = USR(0) (in lokatie 0 moet $00 staan)

of zonder regelnummer:

PRUSR(0)

4.5. Statements.

PRINT val 1, ..., val n

Deze statement print de waarde van val 1 ... n op de terminal.

val 1 ... n = getal, expression, string.

Voorbeeld:

```
PRINT 1,2,B
Stel B = 3
geeft 1       2       3   (kolom van 8 karakters)
PRINT 1;2;3
geeft 123
PR "A=";5,"B=";7,
PR "C;";
PR8
geeft A=5       B=7       C=8
```

Voor een carriage return in een print-opdracht wordt een < X-OFF > control-karakter opgegeven.

INPUT val 1, ..., val n

Deze statement geeft een < ? >- en een < X-ON >-karakter aan de terminal waarna gewacht wordt op input-karakters om val 1 ... val n waarden toe te kennen.

val 1 ... n = getal, expression of een symbol dat reeds door het programma is gedefinieerd.

Voorbeeld:

```
INPUT X,Y,Z     stel A = 1, B = 2, C = 3
Gebruiker geeft A,C,B
Dan wordt X = 1, Y = 3 en Z = 2
INPUT X
Gebruiker geeft +1-3+6 0
X wordt 58
```

In een INPUT-statement zonder regelnummer kunnen waarden direct als volgt aan parameters worden toegekend:

Voorbeeld.

```
INPUT X,1,Y,2,Z,3
dan wordt X = 1, Y = 2 en Z = 3
```

Grotere getallenreeksen kunnen met papertape ingelezen worden als ze gescheiden zijn door komma's en afgesloten met een X-OFF voor de laatste carriage return.

LET var = expression

(Het woord LET is niet noodzakelijk.)

Voorbeeld:

```
LET A=B+C
I = 0
LET Q = RND(RND(33)+5)
```

GOTO-expression

Sprong-functie naar regelnummer < expression >.

Voorbeeld:

```
GOTO 1000
GO TO 200+I*10
G O T O X
```

GOSUB

Sprongfunctie naar regelnummer < expression > met terugkeermogelijkheid.

Voorbeeld:

```
GOSUB 100
GO SUB 200+I*10
```

Terugkeer naar de statement volgend op de GOSUB-instructie vindt plaats na een RETURN-statement.

Voorbeeld:

```
RETURN
```

IF-statement

Vergelijken van 2 expressions.

=	gelijk
>	groter
<	kleiner
<=	kleiner of gelijk
=>	groter of gelijk
<> of ><	niet gelijk.

Voorbeeld:

```
IF I>25 THEN PRINT"ERROR"
IF N/P*P=N GOTO 100
IF RND(100)>50 THEN IF I<>J INPUT Q,R
```

END

Programma afsluiten.

REM-comment

Gebruik van programmatekst.

CLEAR

Maakt het geheugen schoon, erg gevaarlijk dus.

RUN

Uitvoeren van programma.

RUN, expression 1, expression 2, ...

Kent de waarde van expression 1 etc. toe aan de eerstvolgende INPUT-statement.

LIST

Geeft listing van het gebruikersprogramma.

Voorbeeld:

LIST	List alles
LIST 75+25	List regel 100
LIST 100,200	List regel 100 tot 200
LIST 500,400	List niets.

4.6. Fouten.

ERROR MESSAGE SUMMARY

0	Break during execution
8	Memory overflow; line not inserted
9	Line number 0 not allowed
13	RUN with no program in memory
18	LET is missng a variable name
20	LET is missing an =
23	Improper syntax in LET
25	LET is not followed by END
34	Improper syntax in GOTO
37	No line to GO TO
39	Misspelled GOTO
40,41	Misspelled GOSUB
46	GOSUB subroutine does not exist
59	PRINT not followed by END
62	Missing close quote in PRINT string
73	Colon in PRINT is not at end of statement
75	PRINT not followed by END
95	IF not followed by END
104	INPUT syntax bad - expects variable name
123	INPUT syntax bad - expects comma
124	INPUT not followed by END
132	RETURN syntax bad
133	RETURN has no matching GOSUB
134	GOSUB not followed by END
139	END syntax bad
154	Can't LIST line number 0
164	LIST syntax error - expects comma
183	REM not followed by END
184	Missing statement type keyword
186	Misspelled statement type keyword
188	Memory overflow: too many GOSUB's ...
211	... or expression to complex
224	Divide by 0
226	Memory overflow
232	Expression too complex ...
233	... using RND ...
234	... in direct evaluation;
253	... simplify the expression
259	RND (0) not allowed
266	Expression too complex ...
267	... for RND
275	USR expects "(" before arguments
284	USR expects ")" after arguments
287	Expression too complex ...
288	... for USR
290	Expression too complex
293	Syntax error in expression - expects value
296	Syntax error - expects ")"
298	Memory overflow (in USR)
303	Expression too complex (in USR)

304 Memory overflow (in function evaluation)
306 Syntax error - expects "(" for function arguments
330 IF syntax error - expects relation operator

Other error message numbers may possibly occur if the interpreter is malfunctioning. If this happens, check the program in memory, or reload it, and try again.

Error number 194 may also occur if TINY BASIC is incorrectly interfaced to the keyboard input routines. A memory dump of the input line buffer may disclose this kind of irregularity.

4.7. Definities.

FORMAL DEFINITION OF TINY BASIC

```
line ::= number statement CR
         statement CR
statement ::= PRINT printlist
              PR printlist
              INPUT varlist
              LET var = expression
              var = expression
              GOTO expression
              GOSUB expression
              RETURN
              IF expression relop expression THEN statement
              IF expression relop expression statement
              REM commentstring
              CLEAR
              RUN
              RUN exprlist
              LIST
              LIST exprlist
printlist ::=
              printitem
              printitem ;
              printitem separator printlist
printitem ::= expression
              "characterstring"
varlist ::= var
            var , varlist
exprlist ::= expression
             expression , exprlist
expression ::= unsignedexpr
               + unsignedexpr
               - unsignedexpr
unsignedexpr ::= term
                 term + unsignedexpr
                 term - unsignedexpr
term ::= factor
         factor * term
         factor / term
factor ::= var
           number
           ( expression )
           function
function ::= RND ( expression )
             USR ( exprlist )
number ::= digit
           digit number
separator ::= , ! ;
var ::= A ! B ! ... ! Y ! Z
digit ::= 0 ! 1 ! 2 ! ... ! 9
relop ::= < ! > ! = ! <= ! >= ! <> ! ><
```

4.8. Voorbeelden.

AN EXAMPLE PROGRAM

```
10 REM DISPLAY 64 RANDOM NUMBERS < 100 ON 8 LINES
20 LET I=0
30 PRINT RND (100),
40 LET I=I+1
50 IF I/8*8=I THEN PRINT
60 IF I<64 THEN GOTO 30
70 END
```

```
100 REM PRINT HEX MEMORY DUMP
109 REM INITIALIZE
110 A=-10
120 B=-11
130 C=-12
140 D=-13
150 E=-14
160 F=-15
170 X = -1
180 LET S = 256
190 REMARK: S IS BEGINNING OF TINY (IN DECIMAL)
200 REM GET (HEX) ADDRESSES
210 PRINT "DUMP: L,U";
215 REM INPUT STARTING ADDRESS IN HEX
220 GOSUB 500
230 L=N
235 REM INPUT ENDING ADDRESS IN HEX
240 GOSUB 500
250 U=N
275 REM TYPE OUT ADDRESS
280 GOSUB 450
290 REM GET MEMORY BYTE
300 LET N = USR (S+20,L)
305 REM CONVERT IT TO HEX
310 LET M = N/16
320 LET N = N-M*16
330 PRINT " ";
335 REM PRINT IT
340 GOSUB 400+M+M
350 GOSUB 400+N+N
355 REM  END?
360 IF L=U GO TO 390
365 L=L+1
370 IF L/16*16 = L GOTO 280
375 REM DO 16 BYTES PER LINE
380 GO TO 300
390 PRINT
395 END
399 PRINT ONE HEX DIGIT
400 PRINT O;
```

```
401 RETURN
402 PRINT 1;
403 RETURN
404 PRINT 2;
405 RETURN
406 PRINT 3;
407 RETURN
408 PRINT 4;
409 RETURN
410 PRINT 5;
411 RETURN
412 PRINT 6;
413 RETURN
414 PRINT 7;
415 RETURN
416 PRINT 8;
417 RETURN
418 PRINT 9;
419 RETURN
420 PRINT "A";
421 RETURN
422 PRINT "B";
423 RETURN
424 PRINT "C";
425 RETURN
426 PRINT "D";
427 RETURN
428 PRINT "E";
429 RETURN
430 PRINT "F";
431 RETURN
440 REM PRINT HEX ADDRESS
450 PRINT
455 REM CONVERT IT TO HEX
460 N = L/4096
470 IF N<0 N=-N
480 GOSUB 400+N+N
483 LET N=(L-N*4096)
486 GOSUB 400+N/256*2
490 GOSUB 400+(N-N/256*256)/16*2
495 GOTO 400+(N-N/16*16)*2
496 GOTO=GOSUB,RETURN
500 REM INPUT HEX NUMBER
501 REM FORMAT IS NNNNX
502 REM WHERE "N" IS ANY HEX DIGIT
505 N=0
509 REM INPUT LETTER OR STRING OF DIGITS
510 INPUT R
520 IF R=X RETURN
525 REM CHECK FOR ERROR
530 IF R>9999 THEN PRINT "BAD HEX ADDRESS
531 REM NOTE ERROR STOP ON LINE 530 (ON PURPOSE!)
535 REM CONVERT INPUT DECIMAL DIGITS TO HEX
540 IF R>999 THEN N=N*16
545 IF R>99 THEN N=N*16
550 IF R>9 THEN N=N*16
```

```
555 IF R>0 THEN R=R+R/1000*1536+R/100*96+R/10*6
559 REM PICK UP NON-DECIMAL DIGIT LETTERS
560 IF R<0 THEN LET R=-R
565 REM ADD NEW DIGIT TO PREVIOUS NUMBER
570 LET N=N*16+R
580 GOTO 510
590 NOTE: DONT NEED END HERE


1000 TO RUN RANDOM NUMBER PROGRAM, TYPE "RUN"
1010 IT WILL TYPE 8 LINES THEN STOP.
1020 TO RUN HEX DUMP PROGRAM, TYPE "GOTO 100"
1030 IT WILL ASK FOR INPUT.  TYPE 2 HEX ADDRESSES
1040 TERMINATED BY THE LETTER X,
1050 AND SEPARATED BY A COMMA.
1060 THE PROGRAM WILL DUMP MEMORY BETWEEN
1070 THOSE TWO ADDRESSES, INCLUSIVE.
1080 EXAMPLE:
1090 :GOTO 100
1100 DUMP: L,U? A13EX,A246X
1110 A13E EE FF
1120 A140 00 11 22 33 44 55 66
1130 IF THE RANDOM NUMBER PROGRAM
1140 IS REMOVED, OR IF YOU TYPE IN
1150 :1 GOTO 100
1160 THEN YOU CAN GET THE SAME DUMP BY TYPING
1170 :RUN,A13EX,A146X
1180 .
1190 NOTE THAT THIS PROGRAM DEMONSTRATES NEARLY
1200 EVERY FEATURE AVAILABLE IN TINY BASIC.


REMARK: TO FIND OUT HOW MUCH PROGRAM SPACE
REM... YOU HAVE LEFT, TYPE:
LET I=0
1 LET I=I+2
2 GOSUB 1
RUN
REMARK: AFTER A FEW SECONDS, THIS WILL STOP
REM... WITH AN ERROR; THEN TYPE:
END
PRINT "THERE ARE ";I;" BYTES LEFT"


REM: TO EXIT FROM TINY BASIC TO YOUR  ONITOR/DEBUGGER,
LET S=256
REM (S AS IN LINE 180 ABOVE)
LET B=0
IF P=6800 THEN LET B=63
REM: B IS SWI OR BRK INSTRUCTION
LET A = USR (S+24,0,B) + USR (0)
REM: THE FIRST CALL STORES A BREAK IN 0000
REM... THE SECOND CALL JUMPS TO IT.
```

```
:
 LIST

10 REM GEBRUIK USR FUNCTIE
20 REM PRINT KAR OP TERMINAL MET SUBR WRT ($72C6)
30 PRINT"GEEF DE DECIMALE WAARDE VAN HET ASCII-KAR OP:"
50 INPUT K
55 PRINT"<";
60 X=0
70 M=29382
80 A=USR(M,X,K)
85 PRINT">"
90 GOTO 50

:RUN
GEEF DE DECIMALE WAARDE VAN HET ASCII-KAR OP:
? 48
<0>
? 49
<1>
? 65
<A>
? 66
<B>
? 10
<
 >
? 13
<
? 64
<@>
?

!293 AT 50
:
```

```
:LIST

2 REM ** T I N Y  B A S I C **
3 REM SET UP ADC, READVALUE AND PRINT IT
5 REM M=$9200,S=$C014
10 M=37376
20 S=49172
25 REM ADC PIA INITIALISEREN, SELECT DDR A,B
30 A=USR(S+4,M+1,0)
40 A=USR(S+4,M+3,0)
45 REM PA0-7 INPUT
50 A=USR(S+4,M,0)
55 REM PB0-3:INPUT PB4-7: MPX OUTPUT ADDR.
60 A=USR(S+4,M+2,240)
62 REM RETURN DDR A,B FOR DATA
65 REM DATA B,CB1
70 A=USR(S+4,M+3,5)
75 REM DATA A,PULSE C2
80 A=USR(S+4,M+1,44)
85 REM END INITIAL.
86 REM START MAINPROGRAM , CHANNEL INPUT
87 PR"CHANNEL: ";
90 INPUT C
92 T=0
95 REM SHIFT 4 BIT LEFT
100 C=C*16
105 REM SET MPX ADDRESS.
115 REM DUMMY READ: START CONVERSION
120 V=USR(S,M)
125 REM READ HIGH ORDER + MPX ADDRESS
130 W=USR(S,M+2)
135 REM READ LOW ORDER.
140 V=USR(S,M)
200 D=V+(W-C)*256
220 PRD,
230 T=T+1
235 IF T/8*8=T THEN PR
250 GOTO 130

:RUN
```

```
CHANNEL: ? 0
2009    2008    2009    2007    2007    2008    2008    2008
2009    2008    2008    2009    2009    2007    2009    2009
2008    2005    2006    2006    2006    1960    1960    1960
1960    1959    1959    1960    1960    1960    1960    1959
1960    1959    1960    1960    1960    1803    1468    1031
597     261     157     73      86      82      62      61
61      61      60      54      52      49      45      41
37      32      28      25      21      16      12      9
10      8       9       8       8       8       8       5
2       1       1       1       1       1       2       1
1       1       1       1       1       1       1       1
1       1       1       1       1       1       1       4095
4095    4095    4095    4095    4095    4095    4095    4095
4095    4095    4095    4095    4095    0       0       1
1       347     746     1462    2005    2430    3061    3577
3829    4045    4046    4047    4046    4046    4047    4046
4047    4047    4046    4047    4066    4067    4068    4068
4067    4067    4067    4067    4095    4095    4095    4095
4095    4095    4095    4095    4095    4095    4095    4095
4095    4095    4080    4079    4080    4079    4080    4084
4086    4086    4086    4085    4085    4085    4086    4086
4086    4086    4085    4086    4085    4086
!0 AT 220
:
```

5. Geheugengebruik.

Location	Significance
0000-000F	Not used by any version of Tiny
0010-001D	Proto version temporaries
001E-001F	1K split RAM flag (Proto) (0200=split)
0020-0021	Lowest address of user program space
0022-0023	Highest address of program space
0024-0025	Program end + stack reserve
0026-0027	Top of GOSUB stack
0028-002F	Interpreter parameters
0030-007F	Input line buffer & Computation stack
0080-0081	Random Number Generator Workspace
0082-0083	Variable "A"
0084-0085	Variable "B"
...	...
00B4-00B5	Variable "Z"
00B6-00C7	Interpreter temporaries
00B8	Start of User program (PROTO)
00C8-00D7	Sphere parameters (not 0020-002F)
00D8-00E2	Unused by standard version
00E3-00FF	Used by TIM
00E3	CR-LF delay in bit times
00EA-$00EB	Bit time delay baudrate TTY
C000	Cold Start entry point (6800)
C003	Warm Start entry point
C006-8	JMP (or JSR) to character input
C009-B	JMP to character output
C00C-E	JMP to Break test
C00F	Backspace code
C010	Line Cancel code

C111	Pad character
C112	Tape Mode Enable flag (hex 80 = enabled)
C113	Spare stack size
C114	Subroutine to read one Byte from RAM to A (address in X)
C118	Subroutine to store A into RAM at address in X
0200	Beginning of User program

6. CPU 6502-4 hardware.

De CPU-card is gebaseerd op het Euronorm-systeem. De pinaansluitingen van de 64-polige connector (TC.1) zijn compatible met de genormaliseerde pinaansluitingen van het PUZZLE-systeem.
De CPU-card is uit de volgende elementen opgebouwd:

1. CPU-6502.
   Deze microprocessor wordt gestuurd door een 1 Mc clock.
   Alle controlelijnen van deze processor zijn op de connector TC.1 uitgevoerd. De 16 adreslijnen worden direct na de processor gebufferd, waardoor het mogelijk wordt om meerdere eurokaarten met genormaliseerde pinaansluitingen aan de CPU-6502-4 te koppelen.
2. RS-232 interface.
   De interface aansluitpinnen zijn zowel uitgevoerd op de V24-connector (TC.2) als op de flatcable connector (TC.3).
   De interface-buffers voldoen aan de RS-232 standaard.
   De interface-software wordt verzorgd door het monitorprogramma in het I/O-device 6530-004.
3. 6530-004.
   De 8 PA-poorten van dit IC zijn uitgevoerd op de flatcable connector (TC.3). De PB-poorten worden gebruikt voor de terminal interface, IRQ en de chip-select van het IC.
   Het ROM-gedeelte van de 6530 bevat de Terminal Interface Monitor (TIM), hierdoor is geen extra software nodig voor communicatie tussen processor-card en terminal.
4. IC-2716.
   Twee van deze Eprom-IC's geven de mogelijkheid voor het permanent opslaan van een programma van 4 Kbytes. Zie hiervoor het software gedeelte.
5. IC-2114.
   Vier van deze RAM-IC's zorgen voor een 2K-werkgeheugen. In verband met de stack en het monitorprogramma is dit werkgeheugen geplaatst van $0000-$07FF.

De terminal interface, de 4K-eprom en de 2K-ram maken van deze CPU-card een volledig zelfstandig systeem. Door de adresbus-buffering is geen extra interface-kaart nodig bij gebruik in een PUZZLE-systeem.

MEMORY-MAP CPU 6502-4.

Voor globale indeling zie bijlage 'MEMORY-MAP'.

EPROM chip-select.

Deze gebeurt d.m.v. een comperator die de adreslijnen 12, 13, 14 en 15 vergelijkt met de schakelaarstanden van IC-switches in IC-4.

Uitgaande van de notatie '0' = schakelaar dicht en '1' = schakelaar open, is de volgende EPROM-indeling mogelijk:

S0	S1	S2	S3	EPROM-lokaties:
0	0	0	0	$ 0000 - $ 1000
0	0	0	1	$ 1000 - $ 2000
0	0	1	0	$ 2000 - $ 3000
ect.				

De 4K-EPROM zit dus steeds aaneengesloten in het geheugengebied. Indien de Eprom-ruimte wordt gebruikt voor de Resident Assembler en de Tiny Basic moet de Eprom geplaatst worden op $ C000-$ CFFF.

S0	S1	S2	S3	EPROM-lokaties:
1	1	0	0	$ C000 - $ D000

RAM-chip-select.

Voor de adressering van de RAM's wordt de PROM 6301 gebruikt. De adressering van de 6301 gebeurt door adreslijnen 10 t/m 15. Omdat de CPU en het monitorprogramma RAM vereisen in page Ø en page 1, zal 1K RAM op $0000-$03FF gezet moet worden.
Dit houdt in, dat adres $ C0 v.d. 6301 de data $ E krijgt.
Voor de tweede K-RAM geldt het volgende:

Adres 6301	Data 6301	RAM-lokaties
$ C1	$ D	$0400-$07FF
$ C2	$ D	$0800-$0BFF
$ C3	$ D	$0C00-$0FFF
etc.		

Alle andere lokaties van de 6301 krijgen dan de data $F.

De belangrijkste geheugenlokaties voor de gebruiker zijn de volgende:

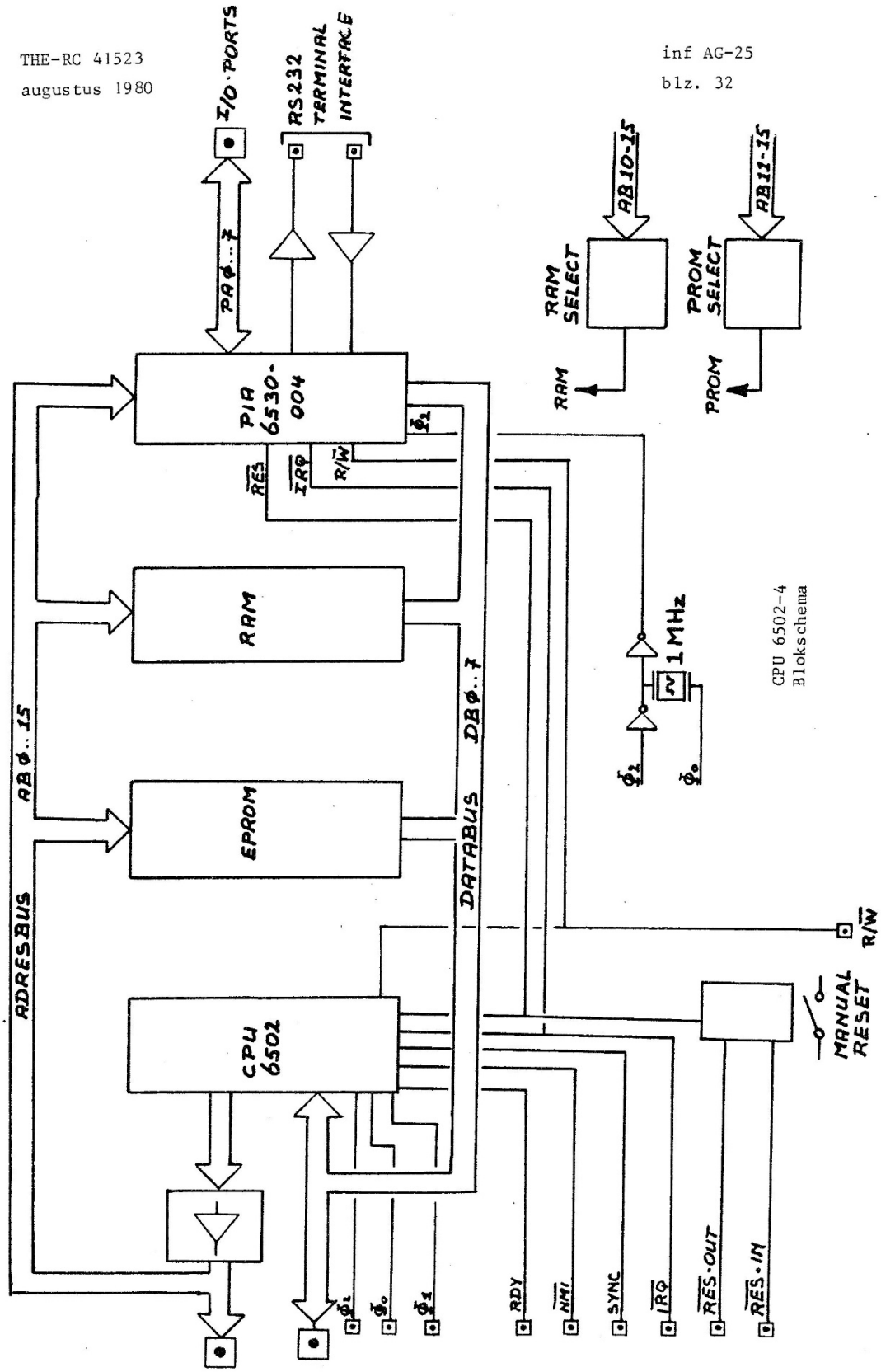
$ FFFE - $ FFFF : IRQ-vector.
$ FFFC - $ FFFD : RESET-vector.
$ FFFA - $ FFFB : NMI-vector.
$ FFF8 - $ FFF9 : UINT-vector: dit is de user IRQ-vector.
$ 7000 - $ 73FF : Interface monitor programma. (ROM-6530)
$ 6200 : PA-poorten 6530.
$ 6201 : DDRA (Data Direction Register A).
$ 6202 : PB-poorten.
$ 6203 : DDRB.
$ 6304 - $ 6307 : Interval Timer 6530.
$ FFC0 - $ FFF7 : User RAM 6530.

Interrupts.

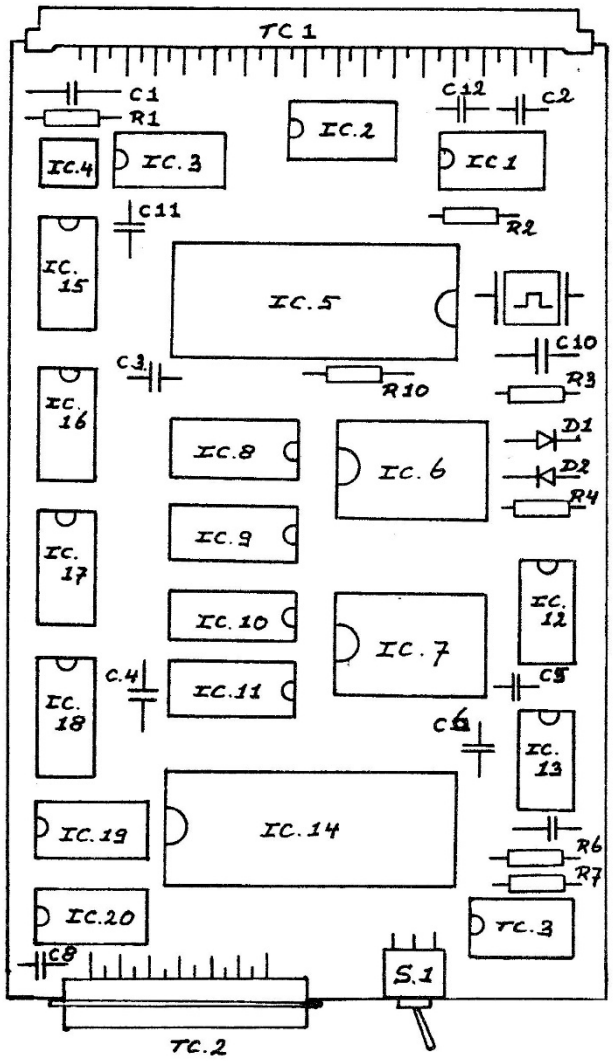
De interrupt-vectoren in de lokaties $ FFFA - $FFFF worden bij het opstarten van de CPU door de monitor geïnitialiseerd. Indien er een IRQ optreedt, dan springt de monitor naar de UINT-vektor, indien deze door de gebruiker niet is geïnitialiseerd, dan handelt de processor deze interrupt af als een NMI.

THE-RC 41523
augustus 1980

inf AG-25
blz. 32



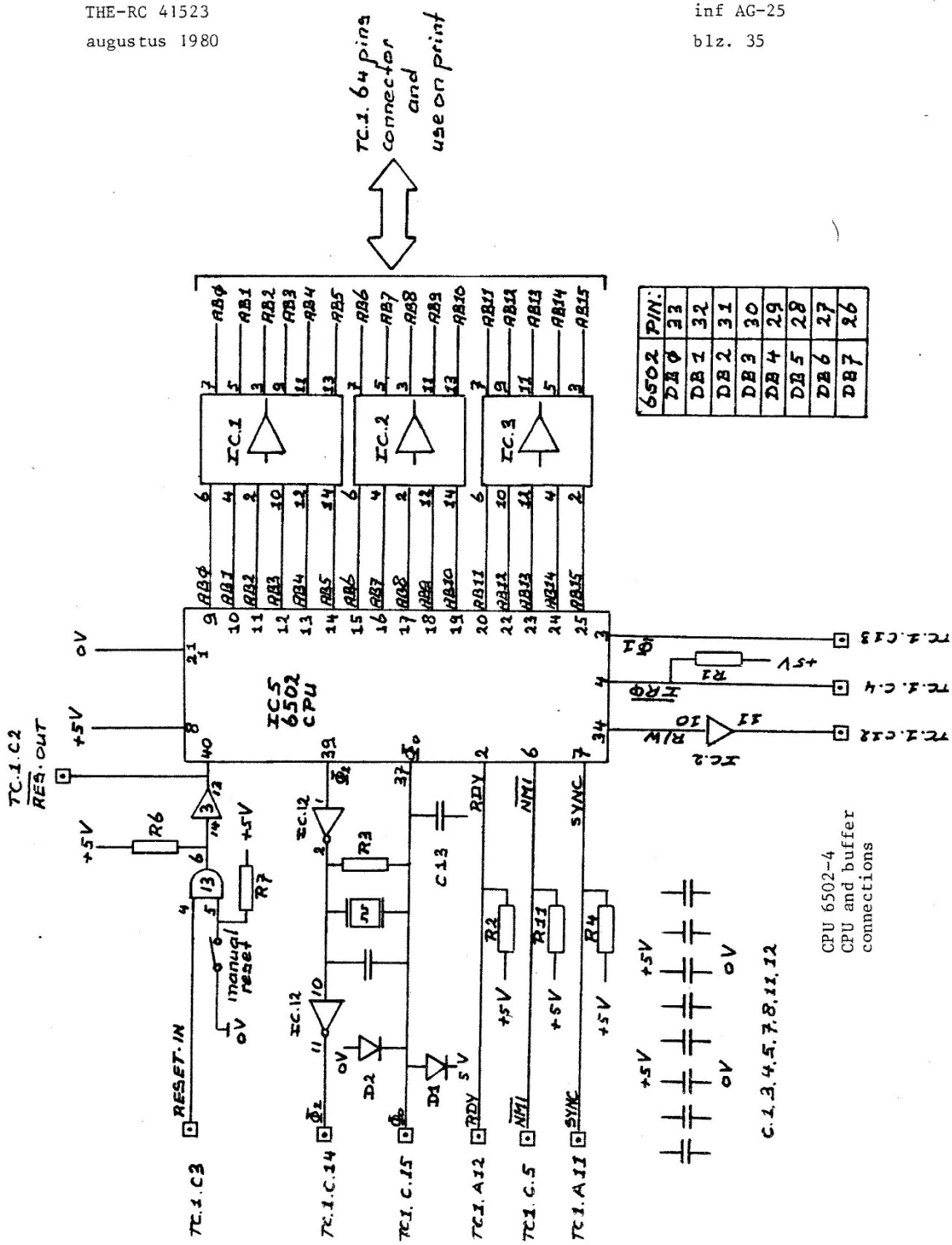
CPU 6502-4
Blokschema



CPU 6502-4
Printindeling

ONDERDELEN CPU-6502-4.

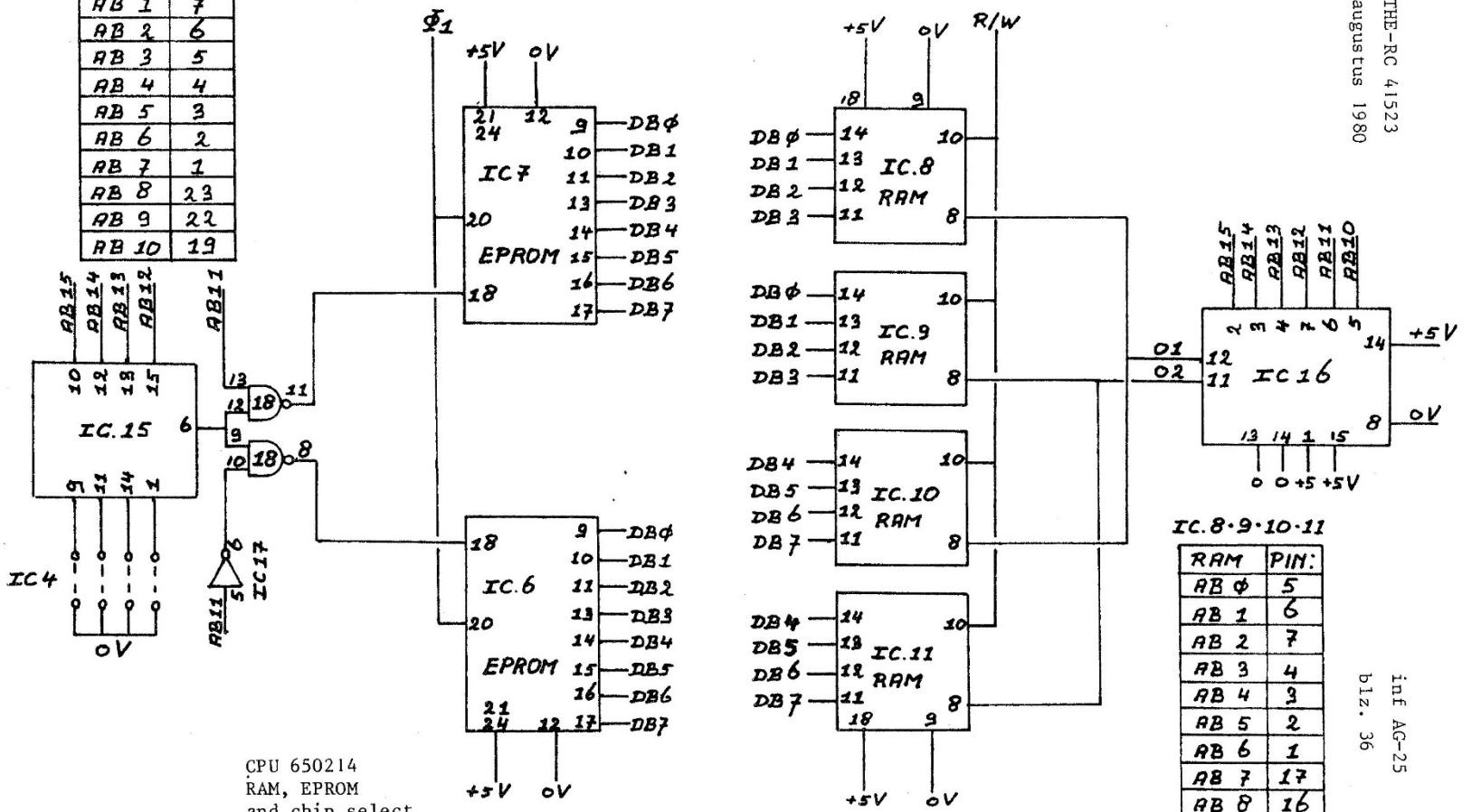
IC 1, 2, 3	74LS365	Buffer
IC 4	IC-switch	4 schakelaars
IC 5	MCS 6502	CPU
IC 6, 7	2716	EPROM
IC 8, 9, 10, 11	2114	RAM
IC 12	74LS04	INVERTOR
IC 13	74LS09	AND
IC 14	6530-004	TIM
IC 15	74LS85	Comperator
IC 16	6301	PROM
IC 17	74LS04	INVERTOR
IC 18	74LS00	NAND
IC 19	9616	Line Driver
IC 20	9617	Line Reciever
R 1, 2, 4, 6, 11	3K3	1/8 W.
R 3	560K	1/8 W.
R 7	1K2	1/8 W.
C 1, 3, 4, 5, 8, 11, 12	100 nF	50 V
C 2, 6	470 nF	10 V
C 10	10 pF	50 V
C 7	10 pF	50 V
TC.1	64 p. con ector	
TC.2	25 p. V24 connector	
TC.3	14 p. flatcable conn.	
Kristal	1 MHz	
D 1, 2	1 N 914	Diode
S 1	Printschakelaar	



CPU 6502-4
CPU and buffer connections

IC. 6·7

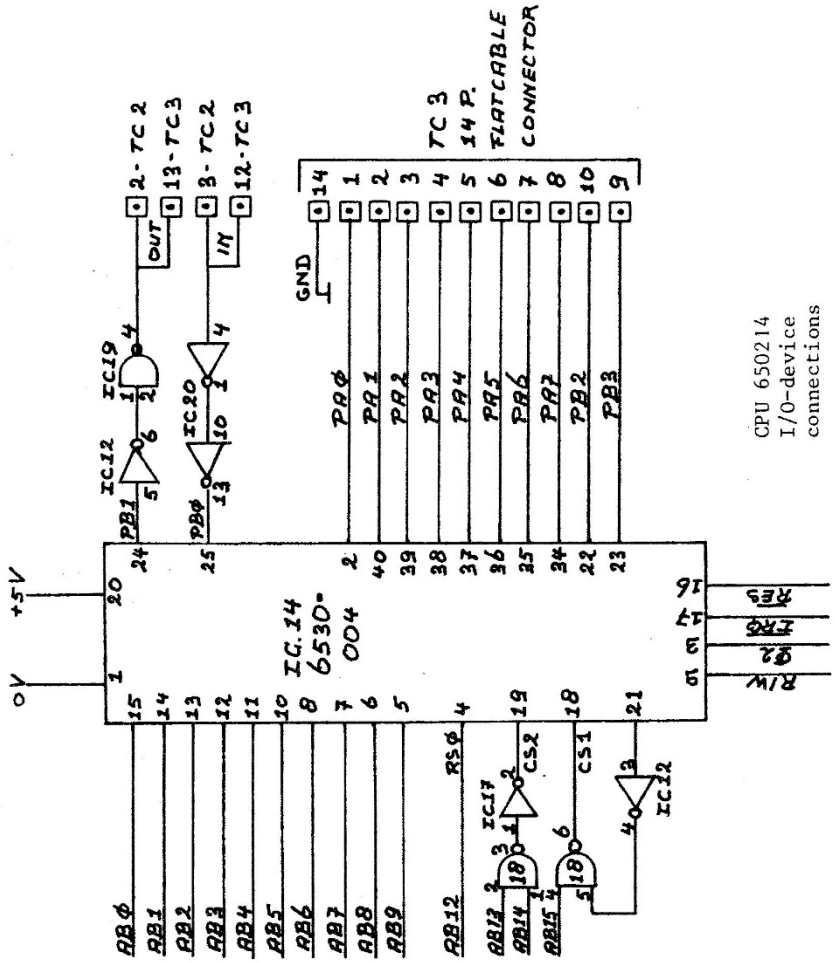
EPROM	PIN
AB Φ	8
AB 1	7
AB 2	6
AB 3	5
AB 4	4
AB 5	3
AB 6	2
AB 7	1
AB 8	23
AB 9	22
AB 10	19



CPU 650214
RAM, EPROM
and chip select
connections

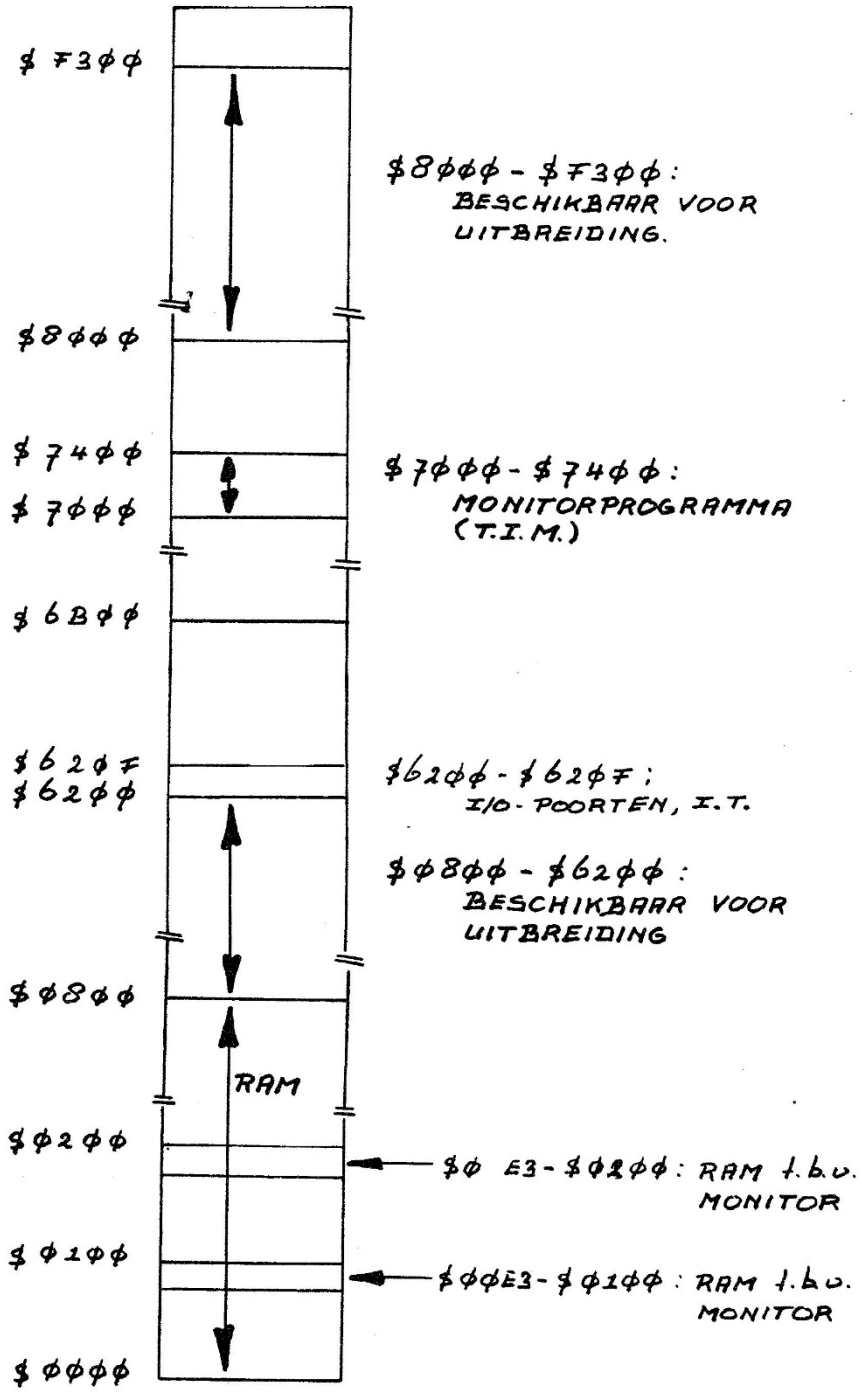
IC.8·9·10·11

RAM	PIN:
AB Φ	5
AB 1	6
AB 2	7
AB 3	4
AB 4	3
AB 5	2
AB 6	1
AB 7	17
AB 8	16
AB 9	15



CPU 650214
I/O-device
connections

6530	PIN:
DB0	33
DB1	32
DB2	31
DB3	30
DB4	29
DB5	28
DB6	27
DB7	26

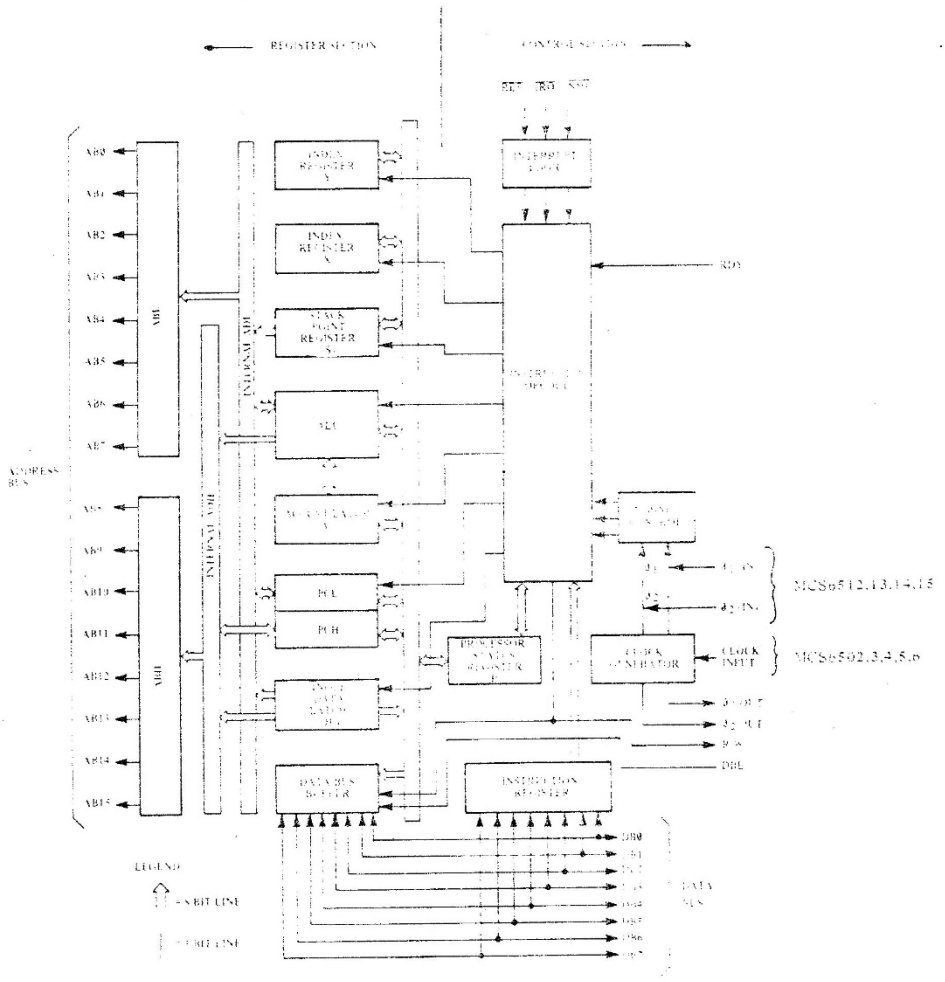


Memory map
CPU 6502-4

Comments on the Data Sheet

The data sheet is constructed to review first the basic "Common Characteristics" - those features which are common to the general family of microprocessors. Subsequent to a review of the family characteristics will be sections devoted to each member of the group with specific features of each.

COMMON CHARACTERISTICS



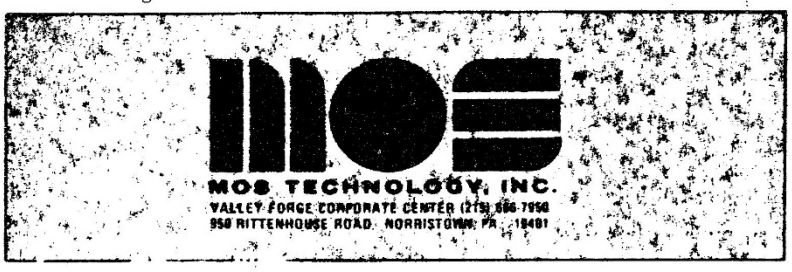
MC66500 Internal Architecture

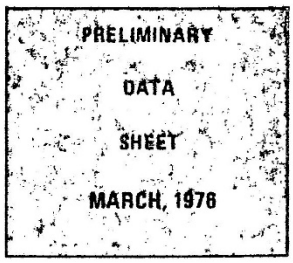
THE-RC 41523

augustus 1980

inf AG-25

blz. 40





MCS6530 (MEMORY, I/O, TIMER ARRAY)

The MCS6530 is designed to operate in conjunction with the MCS650X Microprocessor Family. It is comprised of a mask programmable 1024 x 8 ROM, a 64 x 8 static RAM, two software controlled 8 bit bi-directional data ports allowing direct interfacing between the microprocessor unit and peripheral devices, and a software programmable interval timer with interrupt, capable of timing in various intervals from 1 to 262,144 clock periods.

* 8 bit bi-directional Data Bus for direct communication with the microprocessor
* 1024 x 8 ROM
* 64 x 8 static RAM
* Two 8 bit bi-directional data ports for interface to peripherals
* Two programmable I/O Peripheral Data Direction Registers
* Programmable Interval Timer
* Programmable Interval Timer Interrupt
* TTL & CMOS compatible peripheral lines
* Peripheral pins with Direct Transistor Drive Capability
* High Impedance Three-State Data Pins
* Allows up to 7K contiguous bytes of ROM with no external decoding

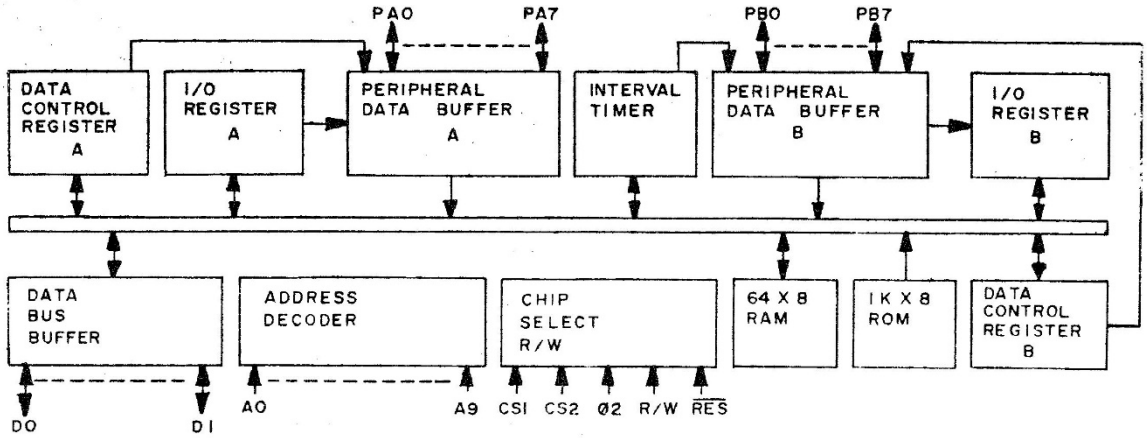


Figure 1. MCS6530 Block Diagram